Air Force Materiel Command

AF Weapon System Software Management Guidebook

AFMC Perspectives

Kevin Stamey
HQ AFMC/ENS
Kevin.stamey@wpafb.af.mil

Presented to 2009 SSTC

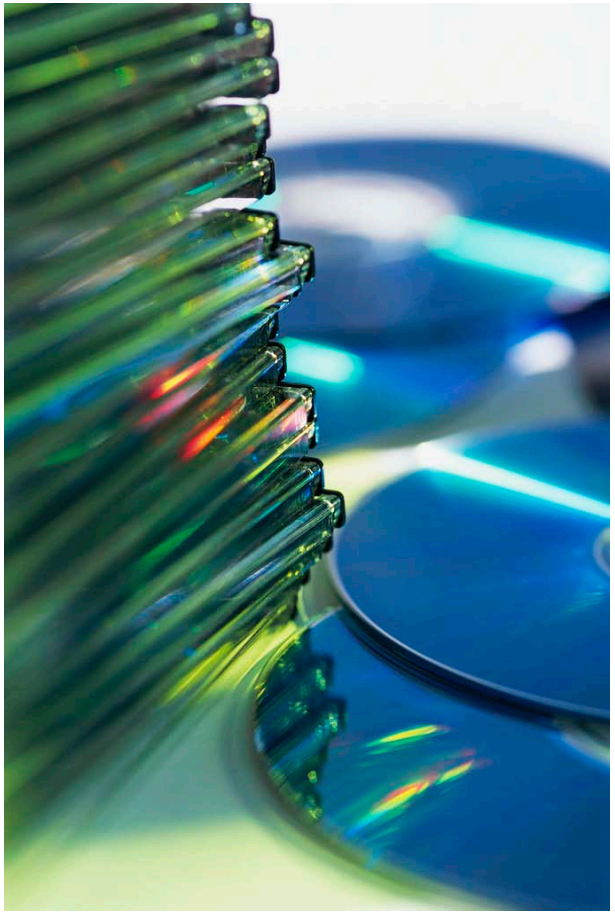
Purpose of AF S/W Guidebook

- **Address persistent problems in S/W acquisition**
  - **Unrealistic baselines**
  - **Evolving requirements**
  - **Insufficient risk management**
  - **Inadequate staff**
  - **Ineffective S/W processes**
  - **Lack of Systems Engineering interface with S/W**
  - **Ineffective management controls/oversight**
- **Recover from atrophy of guidance and policy**

Software Management Guidebook Overview

Software Process Guidelines

- Software Acquisition Planning
- Software Estimating
- Risk Management
- Source Selection
- Earned Value Management
- Requirements Management
- Acquisition insight and involvement
- Safety Critical Software
- Non-Developmental Software
- Software Assurance
- Configuration Management
- Lifecycle Considerations
- Lessons Learned



S/W Acquisition Planning

- **Up-front planning is critical**
  - **Begins with concept exploration**
  - **Builds the foundation to measure success**
- **Prior to contract award**
  - **Conduct independent S/W estimate**
  - **Ensure S/W acquisition strategy is consistent with the system acq strategy including budget and schedule**
  - **Draft Computer Resources Life Cycle Management Plan**
    - **Lifecycle considerations upfront**

Software Estimating

Difficult First Step

- **Realistic S/W cost estimates are necessary to:**
  - **Facilitate realistic overall program planning**
  - **Set reasonable expectations**
  - **Support evaluations of offeror proposals**
- **AF requires 80-90% confidence of estimates**
  - **Poor record of meeting AF requirement**
- **High confidence estimates very difficult**
  - **System performance stated in high level terms**
  - **New systems involve unprecedented systems/capabilities**

High Confidence Estimates

GAO on Software Estimates

"Most often, an *unachievable schedule* causes software cost estimates to be far off target. Playing into this problem is an *overwhelming optimism* about how quickly software can be developed. This optimism stems from a lack of understanding of how staffing, schedule, software complexity, and technology all interrelate. Furthermore, optimism about how much savings *new technology* can offer and the *amount of reuse* that can be leveraged from existing programs also cause software estimates to be underestimated."

GAO Cost Assessment Guide, Exposure Draft
July 2007

High Confidence Estimates

Attributes of "High Confidence" Estimates

- Based upon well defined, stable requirements
- Accounts for confidence to accurately estimate the software size
- Includes appropriate factors for software size growth during development, software reuse, etc.
- Based on realistic (historical) productivity rates
- Actual cost/productivity/SLOC, etc. data is on the same program (or very Analogous Program from same contractor)
- Based on a comprehensive, detailed, realistic, and well documented software development schedule with durations and logic
- A risk assessment, using Monte Carlo simulation, with dollars included in estimate to cover 90% cumulative probability value
- Techniques used are appropriate to the program situation and comprehensiveness of available data
- Independently reviewed and crosschecked at the aggregate level

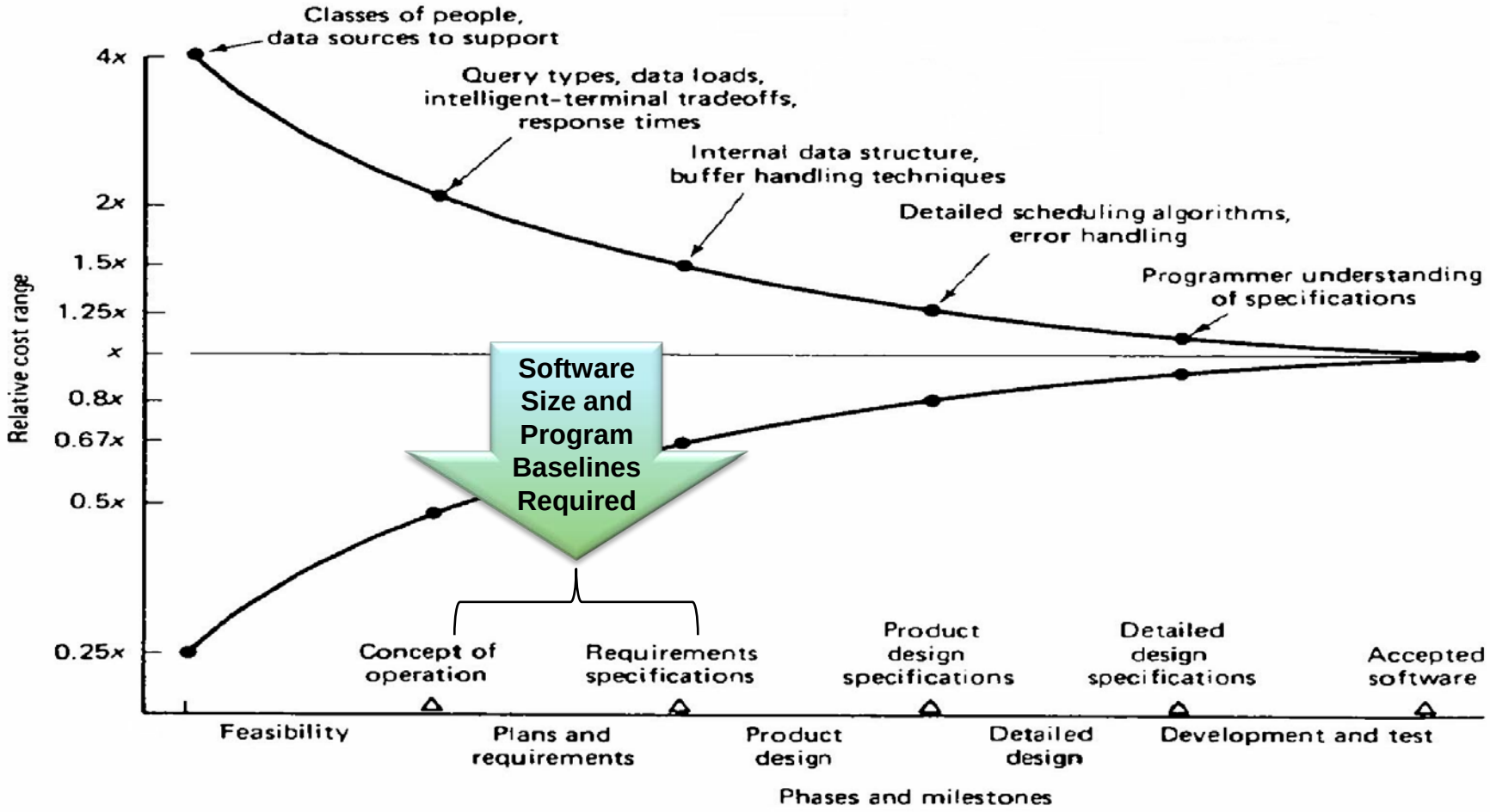
High Confidence Estimates

Risk Factors Affecting Estimates

- **Ability to properly size the software development and integration effort**
- **Ability to account for growth due to developer derived requirements and design evolution (typically 40-50% *or more*)**
- **Ability to achieve planned levels of software reuse**
- **Ability to generate accurate estimates for modifying and/or integrating existing software**
- **Ability to effectively manage requirements and changes, and to accommodate impacts to effort and schedule estimates**
- **Developer capability, including quality of people, processes, plans; and historical productivity**
- **Ability to develop software at predicted productivity rate**

High Confidence Estimates

Software Estimation Accuracy Vs. Phase



From "Software Engineering Economics", Prentice Hall, Barry W. Boehm, 1981

High Confidence Estimates

Software Estimating Process

- **Obtain information on expected software size (KSLOC), complexity, reliability, and other factors**
- **Use commercial estimating models (with cross-checks) to predict the required software effort (cost & schedule)**
  - **Requires numerous assumptions to set model parameters**
  - **Factor in growth**
  - **Account for risks**
  - **Avoid giving undue credit for unproven assertions**
  - **Use Monte Carlo simulations to arrive at confidence factors**
  - **Be cautious of data and assumptions derived from non-similar projects**
- **Develop independent estimate prior to RFP release**
- **Update during source selection**
  - **Use individual offerors' data**
    - **Solicit detailed software size data in proposal**
  - **Support "Most Probable" cost estimate and schedule risk assessment**

High Confidence Estimates

Software Estimating Rules of Thumb

Effort (Cost): Experience On Real-Time Avionics Software Indicates Productivity Is Typically In The Range Of 2-4 Hours Per Source Line Of Code

Schedule: Software Parametric Models Often Have Calibrated Relationships Indicating That The Calendar Time, In Months, Required For Average-Case Software Development Scales Roughly As Five Times The Cube Root Of The Size In KSLOC. An example from Dr. Barry Boehm, USC:

Average-Case Development Time = 5 x Cube Root (KSLOC)

Size (KSLOC)	300	1,000	3,000	10,000
Time (Months)	33	50	72	108

Air Force Experience Suggests This Is A Minimum Schedule

High Confidence Estimates

Software Growth

- **Primary sources of software growth**
  - **Requirements changes/additions**
  - **Improved understanding of the problem**
- **AFMC Study (Early 90's)**
  - **Roughly 25% of changes introduced by government**
  - **Roughly 75% of changes introduced by contractor/developer**
    - ~ 1/3 of these introduced by contractor-derived specification changes
    - ~ 2/3 of these due to evolution of lower-level design requirements as the development proceeds

High Confidence Estimates

Challenges With Software Reuse

- **Development considerations**
  - **Expected reuse often not achieved**
  - **Reuse shortfalls often realized late in development effort**
  - **Expected reuses savings often off-set by additional code development**
- **Long-term support considerations**
  - **Reused software typically coded in legacy languages**
  - **Legacy languages require out-year support**
    - Difficult to find/keep developers experienced in legacy languages
    - Supportable System/Software Engineering Environments (S/SEEs)
    - Declining vendor support for compilers

High Confidence Estimates

Reuse / Size Growth Examples

- **Program 1:**
  - **Initial Program Cost Estimate (PCE) projected $71.2M for EMD**
    - Assumed 18 month development effort / over 80% reuse of existing software
  - **Final PCE projected $145M for EMD**
    - Estimated 60 month development effort / less than 20% reuse of software
  - **Terminated for convenience**

- **Program 2:**
  - **Original software size**
    - 2,048 total KSLOC; 638 KSLOC new/modified; 1,410 KSLOC reused (69%)
  - **Most recent estimate (with 200 KSLOC to go)**
    - 2,156 total KSLOC; 1,108 KSLOC new/modified; 1,048 KSLOC reused (48%)

High Confidence Estimates

Reuse / Size Growth Examples (Cont.)

- **Program 3:**
  - **Original software reuse projection: 75%**
  - **Actual reuse achieved: 25%**

- **Program 4:**
  - Original estimate: 83% reuse
    - 80 KSLOC new; 170 KSLOC modified; 1,225 KSLOC reused (1475 total KSLOC)
  - Current estimate: 61% reuse
    - 720 KSLOC new; 45 KSLOC modified; 1,185 KSLOC reused (1950 total KSLOC)

High Confidence Estimates

An Approach to Reuse Risk and Growth

Attribute	Low Risk Little – No Growth	Moderate Risk Plan for Mod Growth	High Risk Plan for Sig Growth
Maturity	The software is operational in a similar mission application and architecture	The software is in lab or flight test and is not yet operational, or requires architectural adaptation	The software is under development and has not been lab or flight tested
Performance	Performance and interface requirements are verified to be identical	There are some differences in performance and/or interface requirements	There are significant differences in performance and/or interface requirements
Integration Complexity	Integration is not complex and software can be used essentially “as-is” with little or no development	Integration involves minor, localized modifications with moderate complexity	Significant or complex modification or development is required, and/or changes are dispersed
Quality	The software is free of known defects	The software has minor / non-critical limitations or defects	The software has critical limitations or defects that must be corrected
Control	The offeror has full control over the future evolution of the software	The offeror has limited control over the future evolution of the software	The offeror has no control over the future evolution of the software
Access	The offeror has full access to the source code	The offeror has limited access to the source code	The offeror has no access to the source code
Familiarity	The offeror has significant familiarity or experience with the software	The offeror has some familiarity or experience with the software	The offeror has little or no familiarity or experience with the software
Property Rights & Licenses	Adequate property rights & licenses are established	There are uncertainties with property rights / licenses	There are known issues with property rights / licenses
Approach	Proposed approach for achieving predicted reuse is sound, and planned verification is adequate	Proposed approach for achieving predicted reuse is generally sound, with some open issues	Proposed approach for achieving predicted reuse is not substantiated, or planned verification is inadequate
Process	The offeror has an institutionalized process for making reuse decisions	The offeror has an informal process for making reuse decisions	The offeror has no documented process for making reuse decisions

Risk Management

- **Apply "Standard" software risks on all programs:**
  - **Software size and growth (drives effort and schedule)**
  - **Software reuse erosion (proposed levels of reuse not achieved)**
  - **Software producibility (not being realized)**
    - **Especially if expect*ed* productivity significantly higher than historical experience**
- **Identify and manage other software-related risks**

Assume "Standard" Risk Will Occur

Risk Management

Thoughts From Tom DeMarco

"There are many individuals and activities within a project organization whose focus is to reinforce what is going right in a project; risk management's job is to point out what can go wrong. therein lies the problem. Managers and staff who are committed to making the project succeed and who are working long hours under intense pressure do not want to be reminded that one crisis can lead to another. Senior management, customers and stakeholders often adopt the Nike philosophy: just do it! their interests lie in minimizing the reality of risk, not in embracing the fact that risk is a normal part of all projects that cannot be ignored. Risk management hands them more reality than they want to deal with."

Tom DeMarco

Principal of the Atlantic Systems Guild &

Noted Software Author, Educator, and Consultant

Risk must be managed not avoided

Risk Management

Typical Software-Related Risks

1. **Incompatible development performance, effort and schedule**
2. **Rapid staff buildup at the start of new development programs**
3. **Complex, poorly defined, incomplete, or unstable system or software requirements**
4. **Hand-off of software requirements from Systems Engineering without adequate interaction**
5. **Inability to agree on and control build or spiral content (AKA, Lack of a baseline)**
6. **COTS/GOTS availability, suitability, integration, and sustainment**
7. **Integration-heavy effort (significant integration effort for existing components)**
8. **Concurrent hardware development or requirements that drive the use of unproven tools or technology**
9. **Extensive security requirements (Multi-level security)**
10. **Unprecedented system and software architectures**

Risk Management

Typical Software-Related Risks (Cont.)

11. **Long-duration development timeframes**
12. **Technical obsolescence of computing architectures and hardware**
13. **Safety-critical requirements**
14. **Uncontrolled, unknown, or untrusted sources of software (foreign developers, open source, etc.)**
15. **Government Furnished Equipment (GFE) with unknown performance capability**
16. **Use of tools, methods, and technologies with which the developer has no previous experience**
17. **Developer attempting to build systems outside their domain of expertise**
18. **Multiple developers and subcontractors teaming to develop complex software intensive systems which must be tailored and integrated into a total system capability**

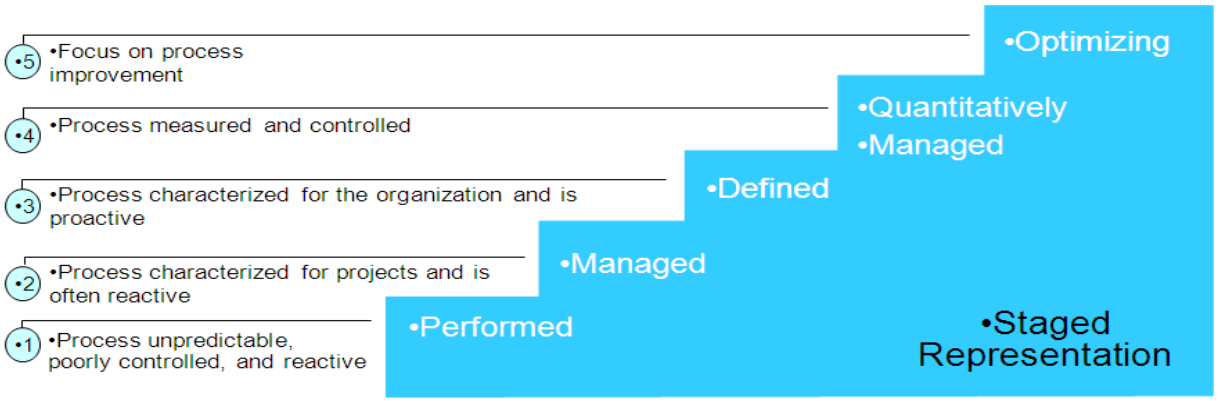
Source Selection

- **Ensure offerors understand the software task**
- **Ensure capability and capacity are consistent with overall software effort**
- **Offer's proposal for Software intensive Systems should be evaluated for:**
  - **Domain experience**
  - **Soundness of approach**
  - **Process maturity**
  - **Past performance with similar S/W efforts**
  - **Program realism**
- **Commitment to proposed process should be evident in S/W Dev Plan, integrated Master Plan and Statement of Work**

Selecting a Capable Developer

What CMM/CMMI Levels Tell Us (or not)

- **Levels are good indicators of *potential organizational performance***
- **They describe how the next project *could perform* based on a sampling of existing projects**
- **Capability Levels and Maturity Levels Reside At the Organizational Level (Corporation, Major Division) and Are Not An indication of How Any individual Project *Is Performing***



- Source: SEI

Capable Developer
Policy on Levels

- **There Is No OSD Or Air Force Policy Requiring A Minimum Maturity Level To Do Business**

  "*DoD does not place significant emphasis on capability level or maturity level ratings*, but rather promotes CMMI as a tool for internal process improvement. This lack of emphasis on ratings is prudent in the light of findings that not all suppliers are exhibiting behavior consistent with their attained CMMI maturity level rating."

Mark Schaeffer & Kristen Baldwin
Defense AT&L: July-August 2007

CMMI as a Tool

- **Specify requirements for process adherence to important process areas**
  - **Use Section H clause to allow government to conduct process reviews**
  - **Gain insight to how processes have been implemented on the program**
- **CMMI results can be requested**
  - **Full Appraisal Disclosure Statement for assessments accomplished in the last 3 years**
  - **Also check SEI's Published Appraisal Results Site**

Evaluating Capability Based on *Past Performance*

Consider the contractor's success in:

- **Planning a software development, integration, and testing effort that had compatible cost, schedule, and performance baselines**
- **Delivering expected software driven capabilities on cost and on schedule**
- **Staffing with the domain expertise and software knowledge, skills, and abilities needed to execute the contract across the lifecycle**
- **Achieving software assurance**
- **Consistent application of documented software engineering and management processes, including technical reviews, in alignment with contract requirements**

Source Selection

Proprietary Software and Rights in Data

- **Identify proprietary software at the start of development**
- **Needed rights are dependent on support approach**
  - **COTS brings special problems**
- **Government must acquire rights to modify Operational Software to be supported organically, by a third party, or through public/private partnerships**
- **Software developed by the program**
  - **Historically - unlimited rights**
- **Other software - negotiate adequate rights**

Earned Value Management

- **Software development produces measurable products**
  - **EVMS should be applied to manage and status S/W development**
- **Identify software through the WBS to the work package level**
  - **Work packages are the source of schedule and cost reporting data**
- **Place software in the WBS consistent with location of software in the systems architecture**
  - **May need special reporting requirements if software is too low in the WBS (avoid roll-up / lost visibility)**
- **Determine software status by measuring technical progress and earned value at the work package level**

Earned Value Management
Approach (Cont.)

- **Solicit Work Package Format in the RFP and Require government access to work packages**
- **Become familiar with the contractor's EVMS system**
- **Understand method used to take earned value in software work packages**
- **Work packages must include activities with defined events**
  - **Objectively verifiable products**
  - **Established earned value**
  - **Schedule**
- **Use earned value system to provide objective insight into cost and schedule performance**
  - **Not at aggregate level**

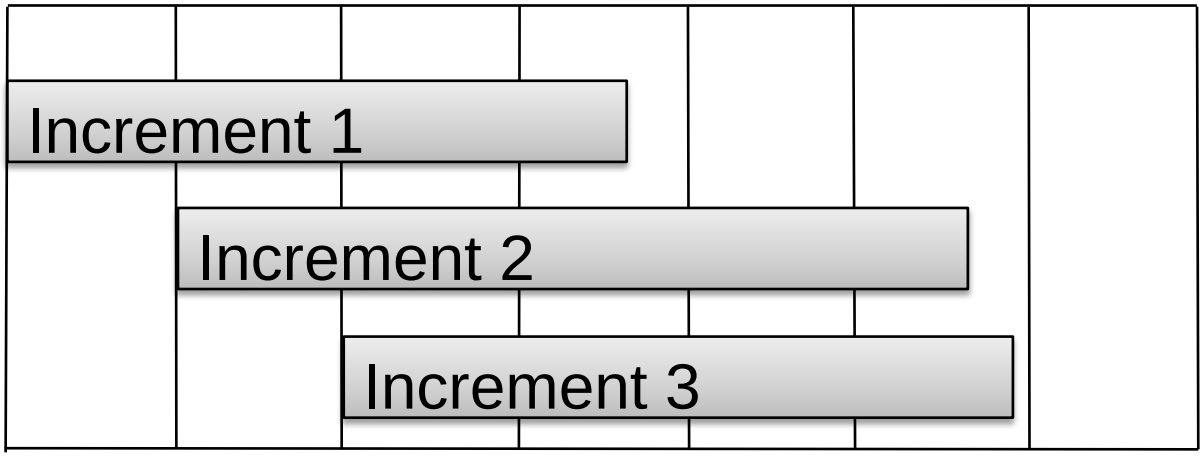
Requirements Management

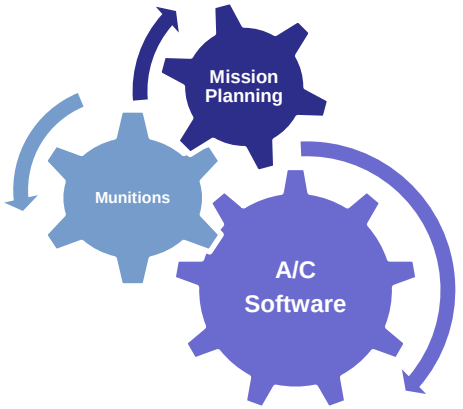
- **Requirement basics**
  - **Single and verifiable statements**
  - **Vertical traceability to capability requirement**
  - **Horizontal traceability to ensure complete functionality**
  - **Allocated to a specific configuration item**
  - **Ensure derived requirements are necessary, sufficient and consistent with system requirements**
  - **Highly recommend use of RM tools**
- **Requirements analysis should evaluate:**
  - **Completeness**
  - **Feasibility**
  - **Trade space**
  - **Verifiability/Testability**
  - **Human factors**
  - **System and SoS architecture**

Requirements Management

During Incremental Dev

- **Approved H/W & S/W interface specifications**
- **Allocate requirements into pre-planned builds**
  - **Expectation management**
- **When two or more concurrent increments are being developed**
  - **Adequate controls required to prevent divergence**
  - **Ensure adequate resources for each increment**





Size Control

Objective of Size Control:

- **Prevent uncontrolled growth in software development**
- **Validate new requirements as necessary to meet overall system performance requirements**
- **Avoid related growth in computer processing resources**
- **Base decisions on cost, schedule and performance impacts**

Plan For Size Control

- Design system/software architecture for inevitable growth
- Plan for growth as requirements are decomposed and derived requirements emerge
  - Validate necessity against system requirements
  - Manage growth to the planned growth estimates
- Defer functionality not essential to system requirements, unless it was part of the plan
- Use formal ECP to adjust cost and schedule

Acquisition insight & involvement

- Identify trends in measured performance
  - Software Metrics
- Proactively identify and manage risk
  - Quantitative and Qualitative indicators
  - Monitor/Track defect and problem reports to closure
- Address problems as they arise
  - Active participation in program reviews
  - Consider participation in peer review, integration testing

Metrics

- **Objective: Quantitative insight into developer activities**
  - **Mutually agree on and implement elected software metrics**
  - **Add unique metrics, as needed**
- **Software metrics information should be available, ideally, through on-line, electronic means**

Air Force Core Metrics

- Software Size
- Software Development Effort
- Software Development Schedule
- Software Defects
- Software Requirements
- Software Staffing
- Software Progress
- Computer Resources Utilization

Metrics

Effective Metrics Characteristics

- **Are integral to the program and directly related to the software development process**
- **Address product, process, and project**
- **Include realistic, achievable planned profiles**
- **Include trend data and management action thresholds**
- **Are actually used to manage the program**
  - **Are collected, analyzed, reported**
  - **Drive further assessment/decision/action when warranted**
- **Are clearly understandable**
- **Clearly portray variances between planned and actual performance**

Metrics

Effective Metrics Characteristics (Cont.)

- **Support the assessment of the impact of proposed changes on the program**
- **Are used (in the form of past actuals) when planning new efforts**
- **Evolve and change when necessary**

- For Metrics Examples:

- See “Practical Systems and Software Measurement” Guidebook
  - http://www.psmsc.com/psmguide.asp

Safety-Critical Software

Special Considerations

- **System safety hazard analysis**
- **Isolation/partitioning of flight-critical computer resources from other processing elements or functions**
  - **If a portion of a Computer Software Configuration Item (CSCI) is designated as safety critical, the entire CSCI is safety critical**
  - **If the software is safety-critical, the associated hardware and system is safety-critical**
- **Fault-containment approaches**
  - **Redundancy, extensive monitors, N-version programming, etc.**
- **Evaluation / verification methodology & discipline**
  - **Structural coverage analysis, safety critical function thread testing, Failure Modes Effects Testing (FMET), etc.**

Safety-Critical Software (Cont.)

Testing

- **Safety-critical software requires testing, for each build/increment, from the lowest level to system level integration**
  - **Test the lowest level of software design at the Computer Software Unit (CSU) level**
  - **Test functional groupings of CSUs and interdependencies at the Computer Software Component (CSC) level**
  - **Test all software requirements at the CSCI level**
  - **Test the integration of the CSCIs at the Operational Flight Program (OFP) level**
  - **Address the integration of hardware and software in a representative integration lab at the System Level Testing (SLT) level**
  - **Insert failures into the actual hardware, software, and system during Failure Modes Effects Testing (FMET)**
  - **Ensure changes to previously delivered safety-critical software are fully qualified**

Non-Developmental Software

- **Robust systems engineering essential**
  - **Risk must be identified and managed**
    - **Security and assurance critical for NDS**
    - **Market forces are the drivers not program needs**
  - **NDS must be analyzed, tested and integrated thoroughly**
    - **Known and unknown defects**
    - **How/When defects are corrected**
  - **Consider lifecycle issues warranties, licenses, update requirements, long term support**
  - **Open Source Software**
    - **Limited experience with DoD embedded systems**
    - **Be aware of OSS licensing requirements**

Software Assurance

- **S/W must be designed to meet performance and security goals in spite of vulnerabilities to:**
  - **Cyber attacks, insiders, physical attacks etc**
  - **Threats that can exist during any phase of the lifecycle**
- **Software Assurance is justified confidence that:**
  - **Software functions as intended**
  - **Free of exploitable vulnerabilities, either intentionally or unintentionally designed or inserted**
- **Measure of confidence is achieved through:**
  - **Planned, systematic set of multi-disciplinary analysis**
- **Software assurance utilizes specific technologies and processes to manage the risk of exploitable vulnerabilities**

Software Assurance & Anti-Tamper (Cont.)

- **Anti-Tamper: Prevents reverse engineering and exploitation of military critical software technologies in order to deter:**
  - **Technology transfer**
  - **Alteration of system capability**
  - **Development of countermeasure to U.S. systems**
- **Anti-Tamper includes processes, activities, and material implementations**
- **Guidebook identifies Anti-Tamper process steps required by DoD policy and Milestone B criteria**

Configuration Management

- **Systems Config Management Plan (CMP) should address S/W unique CM such as**
  - **Approval authorities for software related changes**
  - **Disposition/tracking defects and deficiency reports**
  - **Version control for pre/post release of code**
  - **Release management to fielded assets**
- **Establish what is under configuration control**
  - **Including CSCI, tools, equipment, labs, documentation**
- **Establish when software products will be under configuration control by the government**
- **Consider the use of Automated Computer Program Identification Number System (ACPINS)**

Lifecycle Support

- **Software support planning**
  - **Begin early in acquisition**
  - **Document and coordinate with relevant stakeholders**
- **Support planning based on Source of Repair Assignment Process (SORAP)**
  - **Included sustainment organization up front**
  - **Indentify and agree on future roles/responsibilities**
  - **Establish transition criteria and timing**
  - **Ensure sustainment facilities plans match transition planning**
  - **Procure equipment, labs and tools as required**
  - **Ensure contractor developed labs/tools are documented to support transition**
- **Establish a strategy to respond to hardware and software obsolescence**

Life Cycle Support (Cont)

- **Acquire necessary engineering data**
  - **Minimum set of data support the full lifecycle**
  - **Get concurrence from sustainer**
- **Post Deployment Software Support**
  - **Identify required Systems/Software Engineering Environments for the complete software effort**
    - Development and test tools
- **Consider long term viability of computer system architecture**
  - **Ease of refresh or technology insertion**
- **Long term viability of selected language(s)**
  - **Availability of programmers**
  - **Compiler and tool support**

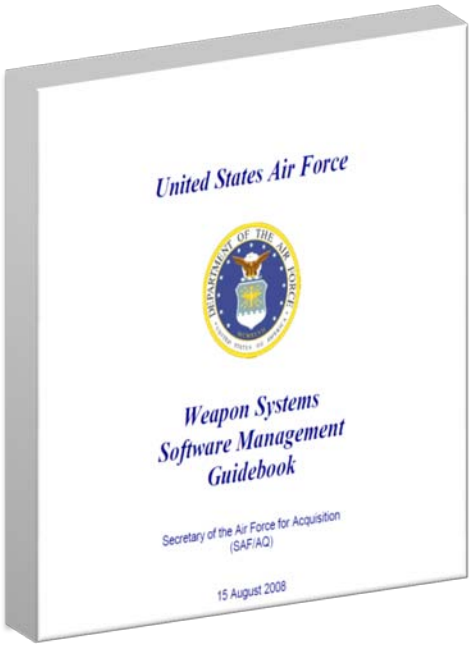
Lessons Learned

- Guidebook is based on proven lessons learned
- Use available resources to benefit from lessons learned
- Contribute to lessons learned repository
- Address:
  - Estimates Vs. Actuals for software size, effort, and schedule
  - Program risks and mitigation approaches
  - Functional requirements changes/additions
  - Schedule perturbations
  - Other program events that contributed to successes and challenges

Get The Guide

- **AF Weapon Systems Software Management Guidebook**
  - **Available on the AF Portal at the Science, Technology & Engineering Functional Home Page**

https://www.my.af.mil/gcss-af/USAAF/AFP40/d/1075546731/Files/editorial/WeaponSystemsSWManagementGuidebook.pdf




LEADING EDGE TECHNOLOGY
WORLD CLASS SUSTAINMENT
WAR-WINNING SYSTEMS